Architetture dei Sistemi   
di Elaborazione

Nome e Cognome ………Simone Pio Candido………………

Matricola …………S337165………………

Il codice compila senza errori: sì [x] no [ ]

Ho provato il progetto in emulazione: sì [ ] no [x]

Ho provato il progetto su board: sì [x] no [ ]

Desidero ritirarmi [ ]

Sistemi basati su ARM T1 – 28 Gennaio 2025

Leggere con attenzione:

1. Occorre sviluppare un progetto ARM usando l’IDE KEIL µVision.
2. Effettuare login su propria area al LABINF ed usare il software disponibile per editare, compilare e debuggare il codice.
3. Utilizzare l’area desktop sul computer del LABINF per creare il vostro progetto.
4. Utilizzare la scheda LANDTIGER o l’emulatore con tutte le non-idealità abilitate per debuggare il progetto
5. Sono inibiti tutti gli accessi ad internet.
6. Si possono utilizzare progetti esistenti, prelevati dalla propria chiavetta USB, ed è possibile consultare materiale cartaceo.
7. Entro l’orario di consegna, occorre finalizzare il salvataggio di tutti i file (valido anche per la parte di modern architecture) e **copiarli nella propria area personale Z:/ all’interno della cartella che contiene le tracce**. Le consegne in ritardo (con file salvati oltre l’orario massimo di consegna) non vengono considerate valide e conducono in ogni caso all’insufficienza.
8. In caso non sia possibile compilare con successo il progetto consegnato, la prova sarà considerata insufficiente. Si richiede di predisporre l’ambiente di debug con le watch che permettono di seguire il flusso del programma.

Esercizio 1 (max 30 points)

Sviluppare le seguenti funzionalità utilizzando le funzionalità della scheda LANDTIGER e del system-on-chip LPC1768.

1. Dichiarare un vettore VETT, composto di LENGTH\_VECT elementi unsigned short, da utilizzare come buffer circolare.
   * Al boot del sistema tale buffer viene resettato, ovvero inizializzato completamente con valori 0; a valle del reset, la prima scrittura avverrà alla posizione 0.
2. Configurare il TIMER 2 in free running mode senza generare interrupt, con un periodo di 100 Hz, ed alimentato da un clock di 50 MHz.
3. Il sistema acquisisce dal JOYSTICK le seguenti posizione (codificate opportunamente), per esempio:
   * UP → 0, DOWN → 1 , RIGHT → 2, LEFT → 3
   * UP-LEFT → 4, UP-RIGHT → 5, DOWN-LEFT → 6, DOWN-RIGHT → 7
4. Il sistema deve mostrare su LED la codifica binaria della posizione del JOYSTICK appena acquisita.

Inoltre, la posizione acquisita deve essere salvata nel buffer circolare implementato con VETT (punto 1).

1. Il sistema deve gestire tre eventi asincroni legati ai bottoni:
   * INT0: se premuto resetta nuovamente il buffer circolare (ovvero riporta il sistema al punto 1). Tutti i LED blinkeranno con una frequenza di 3 Hz, fino alla successiva acquisizione.
   * KEY2: se premuto per più di un secondo estrae i due least significant byte del registro TC (Timer Counter) del timer 2 e lo inserisce nel buffer circolare.
   * KEY1: se premuto per due secondi chiama la seguente funzione assembly (ottimizzata con le istruzioni condizionali):

*int compute\_sum\_and\_ranges (uint\_16\* vett, uint8\_t n, uint8\_t lowerB, int8\_t upperB, int \* rC);*

dove calcola e restituisce la **somma** dei valori del **buffer** circolare *vett* di lunghezza *n* su un intero a 32 bit. Inoltre, la funzione calcola il **numero** di **valori** del buffer circolare che cadono **nel range dei valori** (**compresi**) **lowerB – upperB**, passati alla funzione assembly con valori e scelti a piacere. I valori che cadono nel range [lowerB, upperB] vengono restituiti dalla funzione assembly nella variabile rC passata by reference.

1. A valle dell’esecuzione della funzione assembler:
   * Non sarà possibile acquisire di nuovi valori per i successivi 4 secondi
   * Per 2 secondi - i LED mostreranno l’exor dei 4 byte del risultato dei valori del buffer circolare, lampeggiando alla frequenza di 5 Hz.
   * per 2 ulteriori secondi - i LED mostreranno fisso il valore di rC.
   * dopo i 4 secondi – VETT viene resettato e si riparte dal punto 3.